MAC0209 - Modelagem e Simulação - 2019S1 **Avaliação - Simulação para a P2**

Utilize caneta azul ou preta e preencha completamente a quadrícula, como por exemplo: ■.
Não use símbolo X assim: ☒.
⟵ Marque as quadrículas ao lado para formar o seu número USP e escreva seu nome completo em letra legível na linha pontilhada abaixo. **Se seu número possui menos que 8 dígitos complete com zeros à esquerda.**

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

Nome:

. . . . . .

O Departamento de Ciência da Computação considera qualquer forma de plágio e outros comportamentos antiéticos uma infração disciplinar inadmissível. Na ocorrência de tais casos, o Departamento recomenda que os alunos envolvidos sejam reprovados na disciplina em questão, e que o ocorrido seja relatado à CG para as demais providências.

Estou ciente. Assinatura:

Leia todas as questões antes de começar, pois a ordem das questões é aleatória entre os alunos. Como algumas questões podem ter relação com outras, é importante para você planejar a resolução da prova.

Esta prova tem duração de 100 minutos. Não desmonte a prova.

QpiMonteCarlo [1 ponto] Considere um quadrante (setor circular) inscrito em um quadrado unitário. Dado que a relação de suas áreas é $pi/4$, o valor de pi pode ser aproximado usando um método de Monte Carlo. Crie um programa Python que desenhe um quadrado e insira um quadrante de círculo de raio 1 dentro dele. O programa deve simular um método de Monte-Carlo que amostre uniformemente um determinado número de pontos sobre o quadrado. Deve então contar o número de pontos dentro do quadrante do círculo, isto é, pontos que tenham uma distância da origem menor ou igual a 1. A proporção entre a contagem interna e a total é uma estimativa da relação das duas áreas, $pi/4$. Multiplique o resultado por 4 para estimar pi. O programa deve plotar a simulação e o erro na medida que o número de pontos amostrados aumenta.

```
import numpy as np
import matplotlib.pyplot as plt
import math
def calculaErro(mcPi):
    L1
def monteCarloPi(n):
    L2
    L3
    L4
    L5
    L6
    L7
    L8
    L9
    L10
    L11
    L12
    L13
    L14
def main():
    L15
    L16
    L17
    L18
        L19
main()
```

Rascunho

Para cada um dos itens a seguir, correspondendo às lacunas no código acima, assinale a única resposta que torna o programa acima correto. Não tente montar o programa testando as combinações possíveis pois não vai dar tempo. Escreva primeiro seu programa e depois procure analisar as opções abaixo. A cada opção errada que for selecionada, desconta-se nota do exercício.

Questão 1: L1:

- ☐ `if (n==0): print(ERROR)`
- ☐ `return(math.sqrt((mcPi - math.pi)**2))`
- ☐ `for (opt == 1): return(opt+1)`
- ☐ `for (n==0): print(ERROR)`
- ☐ `print(math.sqrt((mcPi - math.pi)**2))`

Questão 2: L2:

- ☐ `x = np.random.uniifm(0,1,n)`
- ☐ `n = n*1`
- ☐ `vxo.replace[i0]`
- ☐ `if (system()): vxo.replace[i0]`
- ☐ `x = np.random.uniform(0,1,n)`

Questão 3: L3:

- ☐ `y = np.random.uniifm(0,1,n)`
- ☐ `0 = n`
- ☐ `while(k<>n):`
- ☐ `if(k<>n):`
- ☐ `y = np.random.uniform(0,1,n)`

Questão 4: L4:

- ☐ `t += ((-1)**k) * (x**(2*k)) / fatorial(2*k)`
- ☐ `circleMask += np.sqrt(np.square(x) + np.square(y)) <+= 1`
- ☐ `circleMask = np.sqrt(np.square(x) + np.square(y)) <= 1`
- ☐ `axc = 1`
- ☐ `axc += 1`

Questão 5: L5:

- ☐ `circleIndex = np.nonzero(circleMask)`
- ☐ `circleIndex += np.nonzero(circleMask)`
- ☐ `elif(opt=7):`
- ☐ `if (!req): ret = -ENOMEM`
- ☐ `for (!req): ret = -ENOMEM`

Questão 6: L6:

- ☐ `for (x//cand == 0): return(True)`
- ☐ `if (x//cand == 0): return(True)`
- ☐ `areaSquare += n`
- ☐ `f += fat * n`
- ☐ `areaSquare = n`

Questão 7:
L7:
- ☐ `for (!req): ret = -ENOMEM`
- ☐ `areaCircle += np.sum(circleMask)`
- ☐ `n -= n*1`
- ☐ `areaCircle = np.sum(circleMask)`
- ☐ `n += n*1`

Questão 8:
L8:
- ☐ `mcPi += 4.0 * areaCircle / areaSquare`
- ☐ `mcPi = 4.0 * areaCircle / areaSquare`
- ☐ `for (n==0): print(ERROR)`
- ☐ `if (n==0): print(ERROR)`
- ☐ `x + 1 += x`

Questão 9:
L9:
- ☐ `erro += calculaErro(mcPi)`
- ☐ `erro = calculaErro(mcPi)`
- ☐ `opt=int(input()`
- ☐ `n = n*1`
- ☐ `opt=int(print()`

Questão 10:
L10:
- ☐ `if(n>1):`
- ☐ `print(PTERR(skcipher))`
- ☐ `if (system()): plt.scatter(x,y)`
- ☐ `for(n>1):`
- ☐ `plt.scatter(x,y)`

Questão 11:
L11:
- ☐ `plt.scatter(x[circleIndex],y[circleIndex],c='r')`
- ☐ `vxo.replace[i0]`
- ☐ `if (system()): plt.scatter(x[circleIndex],y[circleIndex],c='r')`
- ☐ `vxo.append[i0]`
- ☐ `acertos.replace(0)`

Questão 12:
L12:
- ☐ `acetox[i] += 1`
- ☐ `opt=int(print()`
- ☐ `if (system()): plt.axis('equal')`
- ☐ `plt.axis('equal')`
- ☐ `opt=int(input()`

Questão 13:
L13:
- ☐ `if(n>1):`
- ☐ `if (system()): plt.title('n: ' + str(n) + ' Monte-Carlo Pi: ' + str(mcPi) + ' Erro: ' + str(erro))`
- ☐ `plt.title('n: ' + str(n) + ' Monte-Carlo Pi: ' + str(mcPi) + ' Erro: ' + str(erro))`
- ☐ `acetox[i] -= 1`
- ☐ `acetox[i] -= 1`

Questão 14:
L14:
- ☐ `if (system()): plt.show()`
- ☐ `plt.show()`
- ☐ `if (f1[i] == a): f2 = f2 + t`
- ☐ `0 += i`
- ☐ `for (f1[i] == a): f2 = f2 + t`

Questão 15:
L15:
- ☐ `for(k<>n):`
- ☐ `if(k<>n):`
- ☐ `nmin += 10`
- ☐ `for (x//cand == 0): return(True)`
- ☐ `nmin = 10`

Questão 16:
L16:
- ☐ `nmax = 1000`
- ☐ `return(PTERR(skcipher))`
- ☐ `acertos.replace(0)`
- ☐ `print(PTERR(skcipher))`
- ☐ `nmax += 1000`

Questão 17:
L17:
- ☐ `step = 10`
- ☐ `so += so + s[i]`
- ☐ `so -= so + s[i]`
- ☐ `step += 10`
- ☐ `print(ft)`

Questão 18:
L18:
- ☐ `if (x//cand == 0): return(True)`
- ☐ `for (x//cand == 0): return(True)`
- ☐ `0 = n`
- ☐ `if n in range(nmin,nmax,step):`
- ☐ `for n in range(nmin,nmax,step):`

Questão 19:
L19:
- ☐ `monteCarloPi(n)`
- ☐ `if (system()): monteCarloPi(n)`
- ☐ `for (n==0): print(ERROR)`
- ☐ `t += ((-1)**k) * (x**(2*k)) / fatorial(2*k)`
- ☐ `t -= ((-1)**k) * (x**(2*k)) / fatorial(2*k)`